CUBRID과 Oracle 비교

–

작성년월일: 2009년 3월

목차

1. 일반 기능 비교

1.1 CUBRID 일반 기능 비교

범주	CUBRID	Oracle
테이블 내 속성 수	제한 없음	~1000
DB 이름 길이	최대 17바이트	32 바이트
테이블 이름 길이	제한 없음	~30
속성 이름 길이	제한 없음	~30
레코드 길이	제한 없음	~255000
단일 인덱스의 최대 인덱스 속성 수	제한 없음	~32
BLOB형식 저장소	8바이트의 OID 및 다른 데이터 페이지 또는 외부객체로 저장된 데이터	LOB Segment의 Pointer정보만 저장

1.2 CUBRID 일반 기능 비교

범주	CUBRID	Oracle
데이터베이스 이름	도메인 내에서 고유.	도메인 내에서 고유.
테이블 이름	데이터베이스 안에서 고유. 대소문자 구분 X.	데이터베이스 사용자 계정 안에서 고유.
테이블 이름 제한	반드시 문자로 시작. 특수문자는 _,%, #허용.	반드시 문자로 시작. 특수문자는 _,%, #허용.
예약어/식별자 사용	"" 나 []로 감싸면 사용 가능.	""으로 감싸면 사용 가능

2. 데이터 타입 비교

2.1 CUBRID 데이터 타입-스트링

데이터 형식		설명	크기
Character Strings	CHAR(n)	고정 길이 문자를 지정. n 디폴트 값은 1.	Fixed
	VARCHAR(n)	가변 길이 문자를 지정. n 디폴트 값은 1,073,741,823.	Variable
	NCHAR(n)	고정 길이 national character string을 지정. NCHAR(n) 데이터 타입에 의해 유지되는 string은 지원되는 Character Set 중 하나임.	Fixed
	NCHAR VARYING(n)	가변 길이 national character string을 지정.	Variable
Bit Strings	BIT(n)	바이너리나 16진수 포맷의 고정길이 bit string을 지정. n 디폴트 값은 1.	Fixed
	BIT VARYING(n)	가변 길이 bit string을 지정. n 디폴트 값은 1,073,741,823.	Variable

2.2 CUBRID 데이터 타입-수치형

데이터 형식		설명	크기
Numeric	SMALLINT	16 Bit 정수. UNSIGNED 표현은 지원되지 않음	2
	SHORT	SMALLINT와 동일	2
	INT	32 Bit 정수 UNSIGNED 표현은 지원되지 않음	4
	BIGINT	64 Bit 정수 UNSIGNED 표현은 지원되지 않음	8
	NUMERIC(p,[s])	정밀도(p)와 범위(s)는 명세 가능. 1<=정밀도(p)<=38임. p의 디폴트 값은 15, s는 0.	16
	DECIMAL	NUMERIC과 동일	16

2.2 CUBRID 데이터 타입-수치형

데이터 형식		설명	크기
	FLOAT(p)	정밀도(p)<=7이면 단일 정밀도(32 Bit) 부동 소수로서 표현됨. 8<=정밀도(p)<=19이면 DOUBLE 타입으로 취급됨. 범위는 –10e+38 에서 +10e+38까지 지정 가능.	4
	REAL	FLOAT와 동일	4
	DOUBLE(p)	FLOAT 타입에 명시된 정밀도(p)보다 큼. 범위는 –10e+308 에서 +10e+308.	8
	DOUBLE PRECISION	DOUBLE과 동일	8

2.3 CUBRID 데이터 타입-날짜/시간

데이터 형식		설명	크기
Data-time	DATE	day(일), month(월), year(년)를 표현하는 타입. `DATE 'mm/dd[/yyyy]'`	4
	TIME	hour(시), minute(분), second(초)를 표현하는 타입. `TIME 'hh:mm [:ss] [am | pm]'`	4

	TIMESTAMP	date와 time 조합의 값을 표현(32Bit). 세컨드 단위
		`TIMESTAMP 'hh:mm [:ss] [am|pm] mm/dd [/yyyy]'`
		`TIMESTAMP 'mm/dd [/yyyy] hh:mm [:ss] [am|pm]'`
			4

	DATETIME	date와 time 조합의 값을 표현(64Bit), 밀리세컨드 단위.
		`DATETIME 'mm/dd/yyyy hh:mm:ss[.msec]'`
		`DATETIME 'yyyy-mm-dd hh:mm:ss[.msec]'`
			8

2.4 CUBRID 데이터 타입-집합형

데이터 형식		설명	크기
Collections	SET	중복될 수 없는 값들의 set을 명세	Variable
	MULTISET	중복될 수 있는 값들의 set을 명세.	Variable
	LIST	엔트리에 명세된 순서대로 값들의 set을 명세. 중복 허용.	Variable
	SEQUENCE	LIST와 동일	Variable

2.5 CUBRID vs Oracle 타입 비교 1

Oracle	CUBRID	차이점
BIT	BIT	
BOOLEAN	X	BIT으로 표현
NUMBER(p)	SMALLINT INTEGER BIGINT	smallint ➔ 2byte, integer ➔ 4byte, Bigint ➔ 8byte
NUMBER	FLOAT(n) REAL DOUBLE DOUBLE PRECISION	
NUMBER(p,s)	DECIMAL(p,s) NUMERIC(p,s)	Oracle: 1<=p<=40 , -84<=s<=127 CUBRID: 1<=p<=38 , 0<=s<=p
BINARY_FLOAT	DOUBLE	Oracle: 5 Byte
BINARY_DOUBLE	DOUBLE	Oracle: 9 Byte

2.5 CUBRID vs Oracle 타입 비교 2

Oracle	CUBRID	비고
LOB (BLOB,CLOB,NCLOB) n<=4G	BIT VARYING (n) or Large Object(GLO)	CLOB→varchar로 대체 BLOB→glo로 대체

➔ ODBC/OLEDB 인터페이스에서는 Large Object사용에 제한이 있다.

2.5 CUBRID vs Oracle 타입 비교 3

Oracle	CUBRID	비고
-	TIME	Oracle: 시,분,초 정보 표시 CUBRID: 시간만 표시
DATE	DATE	Oracle: 시,분,초 정보 표시 CUBRID: 날짜만 표시
DATE	TIMESTAMP	Oracle: 시,분,초 정보 표시 CUBRID: 시,분,초 정보 표시 단, CUBRID는 초 단위 연산만 지원
TIMESTAMP	-	Oracle: 10^(-9)단위까지 지원

2.5 CUBRID vs Oracle 타입 비교 4

Oracle	CUBRID	비고
CHAR(n)	CHAR(n)	Oracle: n<=2000 Byte CUBRID: n<=1G
VARCHAR2(n)	VARCHAR(n)	Oracle: n<=4000 Byte CUBRID: n<=1G
LONG(n)		Oracle: n<=2G

3. 함수 및 연산자 비교

3.1 CUBRID vs Oracle 함수 비교 1

Oracle	CUBRID

SYSDATE
 	
연산 단위: 일
 	
`SELECT SYSDATE-1`
`FROM DUAL;`
	SYSTIMESTAMP or SYS_TIMESTAMP or CURRENT_TIMESTAMP
	현재 날짜와 시간을 반환
	
	연산 단위: 초
	
	`SELECT SYSTIMESTAMP-(3600*24)`
	`FROM DB_ROOT;`

-	SYSDATE or SYS_DATE or CURRENT_DATE 현재 날짜를 반환
-	SYSTIME or SYS_TIME or CURRENT_TIME 현재 시간을 반환

➔ TO_TIME, TO_DATE, TO_TIMESTAMP 함수를 사용하여 적절한 타입으로 변환할 수 있다.

3.1 CUBRID vs Oracle 함수 비교 2

Oracle	CUBRID
INSTR	INSTR POSITION
DBMS_RANDOM.VALUE	RANDOM
TO_DATE	TO_DATE TO_TIMESTAMP

3.2 CUBRID vs Oracle 연산자 비교

Oracle	CUBRID
!= ^= < > ÿ=	<>
CONCAT 또는 ||	|| 또는 +
MINUS	DIFFERENCE
INTERSECT	INTERSECTION

3.3 CUBRID vs Oracle 제약조건 비교

Oracle	CUBRID
NOT NULL UNIQUE PRIMARY KEY FOREIGN KEY	NOT NULL UNIQUE PRIMARY KEY FOREIGN KEY
CHECK CREATE TABLE tbl_1(col_1 VARCHAR(10) NOT NULL CONSRAINT(col_2 CHECK(col_1 IN ('male','female'));	-
제약조건의 활성화/비활성화	-

4. 쿼리 비교

4.1 쿼리 변환이 어려운 경우

➔ORACLE에 특화된 기능의 경우 쿼리 변환이 어렵다.

Oracle	CUBRID
CONNECT BY	없음 Stored procedure로 처리
COUNT(*) OVER(analytic function)	없음
With절	없음
MERGE INTO	없음
REGEXP_REPLACE	없음

4.2 표현의 차이를 고려

➔CUBRID와 예약어 차이가 있음을 이해하고 쿼리문을 작성한다.

Oracle	CUBRID
`SELECT month FROM tbl_1;`	`SELECT "month" FROM tbl_1;`

➔CUBRID에서는 ''과 NULL이 다르다.

Oracle	CUBRID
`INSERT INTO tbl_1` `VALUES (123, '');`	`INSERT INTO tbl_1` `VALUES (123, NULL);`

➔CUBRID에서는 != 대신에 <>를 사용한다.

Oracle	CUBRID
`SELECT * FROM tbl_1` `WHERE a != 0;`	`SELECT * FROM tbl_1` `WHERE a <> 0;`

4.3 더미 테이블의 명시

➔ SQL 표준을 적용하여 쿼리를 작성하며, 더미 테이블(db_root)을 명시한다.

Oracle	CUBRID
SELECT 1+1; SELECT 1+1 FROM **DUAL**;	SELECT 1+1 FROM **db_root**;
DELETE tbl_1;	DELETE **FROM** tbl_1;

4.4 데이터 타입의 명시적 정의

➔ CUBRID는 묵시적인 데이터 타입 변환을 지원하지 않으므로, 명시적으로 타입을 정의한다.

Oracle	CUBRID
CREATE TABLE tbl_1(int_column INT); INSERT INTO tbl_1 VALUES('123')	CREATE TABLE tbl_1(int_column INT); INSERT INTO tbl_1 VALUES(123)
SELECT NVL(MAX(int_column), '0') FROM tbl_1;	SELECT NVL(MAX(int_column), 0) FROM tbl_1;
SELECT * FROM tbl_1 WHERE MONTHS_BETWEEN(sysdate, timestamp_col);	SELECT * FROM tbl_1 WHERE MONTHS_BETWEEN(CAST(systimestamp AS date), CAST(timestamp_col AS date));

4.5 집계 함수의 중복 사용

➔ 집계함수를 중복하여 사용할 수 없으므로 서브쿼리를 이용한다.

Oracle	CUBRID
SELECT **SUM(COUNT(**col_1)) FROM tbl_1 GROUP BY col_1	SELECT **SUM(**col_2) FROM (SELECT **COUNT(col_1)** AS col_2 FROM tbl_1 GROUP BY col_1)

4.6 조인 구문 작성

➔ANSI92표준을 적용한 조인 구문을 작성할 것을 권장한다.

1. INNER JOIN 구문:

```
SELECT col_1 FROM tbl_1 t1
   INNER JOIN tbl_2 t2
        ON t1.col_1 = t2.col_2
WHERE t1.col_1 = 'aa' AND t2.col_2 = 'bb'
```

2. OUTER JOIN 구문:

```
SELECT col_1 FROM tbl_1 t1
   LEFT OUTER JOIN tbl_2 t2
        ON t1.col_1 = t2.col_2 AND t2.col_2 = 'bb'
WHERE t1.col_1 = 'aa'
```

4.7 인덱스 힌트/조인 힌트 작성

➔인덱스 힌트: WHERE절 뒤의 USING 구문을 통해 인덱스 힌트를 설정한다.

```
SELECT col_1 FROM tbl_1 t1
WHERE t1.col_1 = 'aa' USING INDEX tbl_1.col_1(+);

SELECT col_1 FROM tbl_1 t1
WHERE t1.col_1 = 'aa' USING INDEX tbl_1.col_1, tbl_1.col_2;
```

➔조인 힌트: SELECT 절에 /*+ */를 사용하여 힌트를 설정한다.

조인 힌트	CUBRID
USE_NL	질의 최적화기는 중첩 루프 조인 실행 계획을 만든다.
USE_MERGE	질의 최적화기는 정렬 병합 조인 실행 계획을 만든다.
USE_IDX	질의 최적화기는 연관된 인덱스 조인 실행 계획을 만든다.

```
SELECT /*+ JOIN_HINT */ col_1 FROM tbl_1 t1
INNER JOIN tbl_2 t2 ON t1.col_1 = t2.col_2
WHERE t1.col_1 = 'aa' AND t2.col_2 = 'bb'
```

4.8 CASE WHEN 구문

➔CUBRID에서는 CASE WHEN구문에 EXIST조건식을 결합할 수 없다.

Oracle	CUBRID
지원되는 조건식:: •비교 조건식 •NULL 조건식 •RANGE 조건식 •IN 조건식 •패턴 매칭 조건식(LIKE) •EXIST 조건식 SELECT col_1 CASE WHEN EXISTS (SELECT col_2 FROM tbl_2 t2 WHERE t2.col_2 = t1.col_2) THEN 'aaa' ELSE 'bbb' END AS status FROM tbl_1 t1;	지원되는 조건식:: •비교 조건식 •NULL 조건식 •RANGE 조건식 •IN 조건식 •패턴 매칭 조건식(LIKE) SELECT col_1 CASE WHEN 1=(SELECT COUNT(1) FROM tbl_2 t2 WHERE t2.col_2 = t1.col_2) THEN 'aaa' ELSE 'bbb' END AS status FROM tbl_1 t1;

4.9 검색 결과의 일부 출력

➔ ORDER BY절에 의한 정렬 이후의 값을 얻기 위해서는 ORDERBY_NUM()함수를 사용한다.

Oracle	CUBRID
SELECT * FROM(SELECT col_1,col_2,ROWNUM rnum FROM tbl GROUP BY col_1) WHERE RNUM between 1 and 10;	SELECT col_1, col_2 FROM tbl GROUP BY col_1 **HAVING GROUPBY_NUM()** between 1 and 10;
SELECT t2.* FROM(SELECT t1.*, ROWNUM rnum FROM (SELECT col_1, col_2 FROM tbl ORDER BY col_1) t1) t2 WHERE rnum between 1 and 10;	SELECT col_1, col_2 FROM tbl ORDER BY col_1 **FOR ORDERBY_NUM()** between 1 and 10;

4.10 검색 결과의 가로 출력(PIVOT 기능)

➔ SYS_CONNECT_BY함수를 대신하여 LIST나 SET 함수를 이용하여 PIVOT 기능을 구현할 수 있다.

1. 정렬이 필요 없이 col_3을 가로 출력:

```
SELECT id
, LIST(SELECT id,col_3 FROM tbl_1 t1 WHERE t2.id = t1.id )
FROM tbl_2 t2;
```

2. col_1, col_2기준으로 정렬한 순서대로 col_3을 가로 출력:

```
SELECT id,
   LIST(SELECT col_3 FROM (SELECT id,col_3 FROM tbl_1 t1
   WHERE t2.id = t1.id ORDER BY col_1, col_2) t3
        on t3.id = t2.id),
FROM tbl_2 t2;
```

4.11 자동 증가 속성을 이용한 구문

➔수치형 컬럼에 대해 AUTO_INCREMENT 을 적용하여 한번에 조회수를 증가시킬 수 있다.

Oracle	CUBRID
CREATE TABLE tbl_1(id INTEGER, text varchar2(4000)); CREATE SEQUENCE id_val START WITH 1000 INCREMENT BY 1;	CREATE TABLE tbl_1 (id INTEGER **AUTO_INCREMENT**(1000, 1), text VARCHAR(4000));
INSERT INTO tbl_1 VALUES ((SELECT id_val.nextval as id FROM dual), 'aaa');	INSERT INTO tbl_1(text) VALUES('aaa');

4.12 조회수 카운트를 위한 구문 작성

➔INCR함수를 이용하여 SELECT와 UPDATE를 한번에 실행하여 조회수를 증가시킬 수 있다.

Oracle	CUBRID
SELECT content, click_cnt FROM board_1; UPDATE board_1 set click_cnt = click_cnt +1;	SELECT content, **INCR**(click_cnt) FROM board_1;

4.13 기타 비교

기능	지원 여부	비고
Prepare Statement pooling	O	하나의 커넥션 ➔ 하나의 결과값 핸들링 가능 (AUTOCOMMIT이 ON으로 설정,디폴트). 하나의 커넥션 ➔ 여러 개의 결과값 핸들링 가능 (AUTOCOMMIT이 OFF으로 설정).
Stored Procedure	O	PLSQL지원 X Java Stored Procedure 지원 O
FULL 외부조인	X	Left, right outer Join은 지원.
Character set	O	캐릭터셋은 NLS를 고려할 필요 없이 AP환경을 그대로 사용함
속성 크기 변경	X	ALTER TABLE구문을 통한 속성 크기 변경 불가.
Temp table	X	
WITH 절	X	

4.14 정리

기능	비고
표현식의 차이점	예약어는 큰 따옴표로 감싸서 사용 더미 테이블(db_root)을 반드시 명시 != 대신에 <>의 사용
묵시적 타입 변환	명시적으로 데이터 타입 지정 및 데이터 입력
집계함수의 중복 사용	서브쿼리를 이용하여 작성
조인/인덱스 구문	ANSI92표준 적용 구문으로 작성을 권유. 지원되는 조인 힌트 확인.
클릭 카운트(INCR함수)	Oracle의 UPDATE 및 SELECT을 한번에 실행
AUTO_INCREMENT	Oracle의 SEQUENCE를 이용한 구문을 한번에 실행
검색 결과의 일부 출력	ORDERBY_NUM() 사용 Oracle의 2단,3단 서브쿼리를 한번에 실행
검색 결과의 가로 출력	SET or LIST 및 조인을 이용하여 대체